Alternativas Hardware para la Locomoción de un Robot Ápodo

González J., González I. y Boemo E.

Escuela Politécnica Superior, Universidad Autónoma de Madrid, España,
{Juan.Gonzalez, Ivan.Gonzalez, Eduardo.Boemo}@ii.uam.es
http://www.ii.uam.es

Resumen. La robótica modular reconfigurable es una alternativa que ofrece gran versatilidad para la locomoción de robots. En este artículo se presenta una unidad de control de robots basada en *FPGA*, en lugar de un microprocesador específico. Para ello se han realizado dos implementaciones de un sistema de control de locomoción para el robot ápodo *Cube Reloaded*, siguiendo dos enfoques distintos: lógica combinacional y secuencial por un lado y una *CPU* empotrada por otro. Se ha restringido el ámbito de estudio a la locomoción estáticamente estable con un desplazamiento en línea recta generado a partir de ondas sinusoidales que recorren el gusano desde la cola hasta la cabeza.

1 Introducción

La locomoción es la capacidad de los robots para desplazarse de un punto a otro. Engloba varios aspectos: desplazamiento de un incremento en línea recta, movimientos en cualquier dirección dentro de un plano, planificación de trayectorias y navegación. La locomoción es especialmente importante en la exploración de terrenos desconocidos. Tradicionalmente se han diseñado robots específicos para desplazarse en entornos conocidos. Las características del medio se fijan como parámetros de diseño del robot. Tal es el caso del CMU Ambler[1,2], diseñado para la exploración de la superficie de Marte, o de Dante II[3,4], utilizado en Julio de 1994 para descender al interior del cráter del Monte Spurr en Alaska. Ambos proyectos fueron subvencionados por la NASA.

En 1994, Mark Yim[5] diseñó y construyó el robot Polypod[6,7,8], aportando un nuevo enfoque al problema de la locomoción: el diseño de robots modulares y reconfigurables, capaces de cambiar su forma para desplazarse de una manera u otra, según el terreno. En 1998 se desarrolló en el Park la primera versión del robot Polybot[9,10], constituido por módulos mecánicamente más simples. Se implementaron diferentes formas de locomoción, consolidando la robótica modular reconfigurable como una opción viable y muy prometedora, aunque todavía lejos de poderse utilizar en aplicaciones prácticas.

Un robot modular reconfigurable está constituido por varias clases de módulos iguales. Cada módulo se puede considerar como un robot autónomo. Tiene todos los elementos necesarios: estructura mecánica, *hardware*, *software* y alimentación. El robot se puede reconfigurar, cambiando su aspecto y utilizar así unos "andares" (*gaits*) diferentes. Para un terreno se puede mover como un gusano y para otro se puede transformar

en una araña. La característica buscada con estos robots es la versatilidad. Al cambiar de forma, se pueden desplazar por terrenos muy dispares. El hardware incorporado tiene que ser también lo más versátil posible. En los módulos de última generación de Polybot (G2 y G3), se utiliza un procesador Power PC 555 con 1MB de RAM, aunque su potencia no está totalmente utilizada.

Existe una alternativa, todavía no explorada, para lograr que los módulos sean más versátiles: que su *hardware* esté constituido por una *FPGA* que también se pueda reconfigurar dinámicamente, generándose el *hardware* necesario para cada ocasión. Si el robot se reconfigura y cambia su forma, parece lógico permitir que su *hardware* también se pueda modificar.

En este artículo exploramos la viabilidad de utilizar *FPGAs* para conseguir la locomoción del robot ápodo Cube Reloaded en lugar de emplear un microprocesador dedicado. Nos restringiremos a la locomoción estáticamente estable y al desplazamiento del robot en línea recta. Se estudiarán dos alternativas: una basada en lógica combinacional y secuencial y otra utilizando una CPU empotrada.

Este trabajo se ha organizado de la siguiente manera. Primero, se presenta el robot ápodo *Cube Reloaded*. A continuación, se describen las herramientas *software* y *hardware* empleadas en los desarrollos. Se muestra el diseño de las unidades *hardware* necesarias para el control de servomecanismos y presentan las dos alternativas para la locomoción de *Cube Reloaded*. Finalmente, se resumen los principales resultados y conclusiones.

2 El robot ápodo *Cube Reloaded*

El robot ápodo Cube Reloaded (figura 1) está siendo diseñado en la UAM como una versión mejorada de Cube[12,13]. La primera versión está constituida por 4 módulos iguales cada uno con un servomecanismo Futaba 3003 muy comunes en aplicaciones de aeromodelismo. En conjunto, los cuatro módulos se asemejan a un gusano. Estos servos se controlan mediante una señal PWM de 50Hz y anchuras de pulsos comprendidas entre 0.3 y 2.3ms. La electrónica y la alimentación se encuentran fuera del gusano. Los módulos están conectados de manera que todos pertenecen al mismo plano, por lo que el gusano sólo puede moverse en línea recta. Para conseguir el movimiento se utilizan tablas de control, igual que en Polypod y Polybot, que son generadas mediante un software de simulación[13] (figura 2) . Cada fila contiene la posición de todos los servos, en un instante de tiempo. Las posiciones de los servos se calculan aplicando ondas sinusoidales que recorren el gusano desde la cola hasta la cabeza. El movimiento se consigue recorriendo la tabla y enviando las posiciones a los servos. Se trata de un método de control sencillo y eficaz, que se puede implementar muy fácilmente en microcontroladores de 8 bits. El control es en bucle abierto. Para el control de Cube Reloaded se ha empleado una tarjeta CT6811[11], que genera las señales *PWM* a partir de la tabla de control que tiene almacenada en la memoria *eeprom* o que recibe desde el PC a través del *RS-232*.

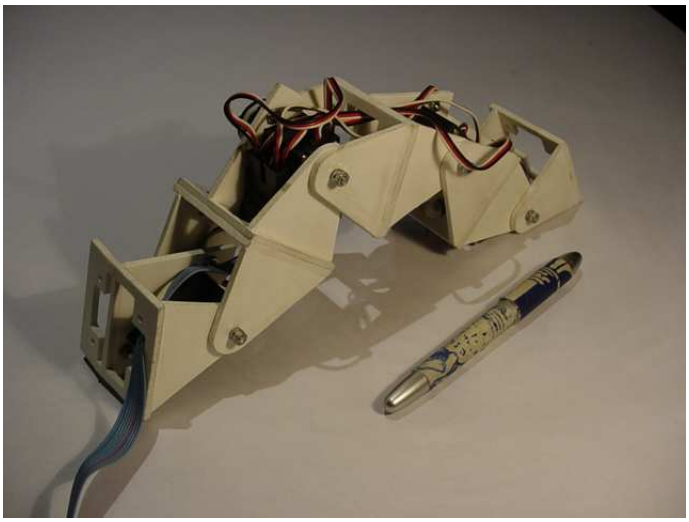

Fig. 1. El robot ápodo Cube Reloaded, constituido por 4 módulos iguales, en cada uno de los cuales hay un servo del tipo Futaba 3003. La alimentación y la electrónica se encuentran situados fuera de la estructura.

3 Tabla de control y señales PWM

Para la locomoción del robot es necesario generar las señales PWM a partir de la información que se encuentra en la tabla de control, tal como se muestra en el cuadro 1. Cada posición de los servos se codifica con 8 bits y en cada fila hay 32 bits. En total la tabla tiene 30 filas.

Instante	Servo 1	Servo 2	Servo 3	Servo 4
0	98	51	6B	9B
1	9D	56	62	9D
2	9B	61	59	9D
...	...	...	...	...

Tabla 1. Aspecto de la tabla de control. Cada fila contiene las posiciones de los 4 servos en un instante de tiempo

Las unidades PWM se ha tomado del proyecto Labobot[18,19] y se han optimizado, usando un único contador de 11 bits para todas ellas. El diseño se puede ver en la figura 3 (“Hardware para PWM”). El contador de 11 bits se incrementa a una frecuencia de 100KHz, volviendo al estado inicial cada 48.8Hz. Se obtienen unas señales PWM

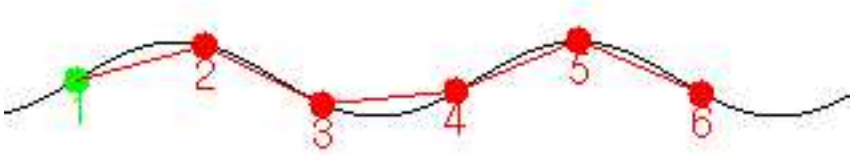

Fig. 2. La generación de movimiento se hace usando ondas sinusoidales que recorren el gusano desde la cola hasta la cabeza. El software desarrollado para Cube calcula las tablas de control en función de los parámetros de la onda (amplitud y longitud de onda)

de frecuencia 48.8Hz, que entran dentro de los márgenes de tolerancia del servo (la frecuencia nominal es de 50Hz). Los comparadores establecen el nivel de salida del PWM, según que la posición deseada sea mayor o menor que el valor del contador.

La plataforma *software* de desarrollo se basa en el sistema operativo Linux y el flujo de diseño se ha realizado empleando la herramienta libre Ghdl[16], basada en el compilador *GCC*, para el compilado/simulado de los diseños en *VHDL*, el sintetizador Synplify Pro 7.1.1 a través de Wine[17] y el entorno ISE 4.2 de Xilinx, también a través de Wine, para la parte de implementación.

La plataforma *hardware* de desarrollo es la tarjeta JPS[14,15] con una *FPGA Spartan I*, y un oscilador de 1MHZ para la señal de reloj. Los *bitstreams* generados se descargan en ella conectándola al puerto serie del PC, a través de un interfaz desarrollado con la tarjeta CT6811 y un software bajo Linux. Los diseños finales se han grabado en una memoria *eeprom* serie (AT17C128), que se inserta en la tarjeta JPS y al alimentar el sistema o pulsar el botón de programación, la FPGA se configura y el gusano comienza a moverse.

4 Alternativa I: lógica combinacional y secuencial

Como se ha comentado en el punto 1, el movimiento del robot ápodo se consigue recorriendo la tabla de control y enviando los valores correspondientes a los motores. Este mecanismo, que inicialmente se había implementado en un microcontrolador de 8 bits, presenta un diseño muy simple en *FPGA*: un contador, una memoria *ROM* y las unidades pwm necesarias. El diagrama de bloques se muestra en la figura 3. Está constituido por dos partes: las unidades de PWM y el secuenciador que devuelve consecutivamente las filas de la tabla de control. Esta tabla se encuentra en una ROM de 32x32, direccionada mediante un contador módulo 30. Mediante un multiplexor de 8 a 1, con sus entradas de selección conectadas a tres *switches* de la tarjeta JPS, se selecciona una

señal de reloj de entre 8 posibles, con los siguientes periodos: 1000, 524, 262, 131, 66, 33, 16 y 8 ms.

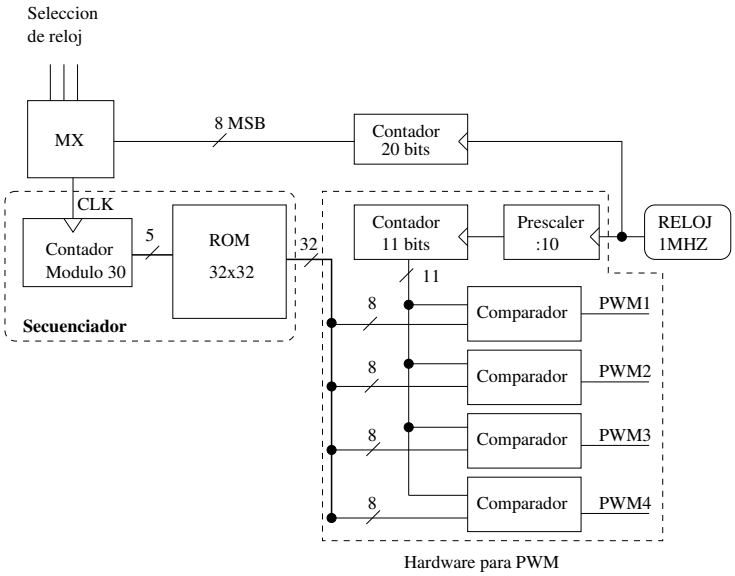

Fig. 3. Diagrama de bloques de la alternativa I.

El periodo de la señal de reloj del secuenciador (T_{CLK}) determina el tiempo que se tardará en enviar la siguiente posición a los servos. Cada servo tarda un tiempo t_{Ser} en alcanzar la nueva posición. Si $T_{CLK} > t_{Ser}$, el servo permanecerá un tiempo $T_{CLK} - t_{Ser}$ en reposo en la nueva posición, antes de que llegue la siguiente. Si esta diferencia es del orden de las centenas de ms, se apreciará un movimiento "a saltos" (no continuo). Por el contrario, si $T_{CLK} < t_{Ser}$, se enviará la nueva posición al servo antes de que haya alcanzado la anterior por lo que los servos nunca llegarán a alcanzar las posiciones indicadas en la tabla de control.

5 Alternativa II: CPU empotrada

El sistema tradicional de control empleado en microrobots se basa en el uso de microcontroladores. Estos sistemas presentan una enorme gama de modelos, diferenciados por la cantidad de recursos/periféricos que los componen. El uso de sistemas basados en microcontrolador permite disponer de la flexibilidad del software, pero en muchas ocasiones, el número de recursos/periféricos es menor al necesario para la tarea a realizar. La combinación de microcontrolador y dispositivo reconfigurable es quizás la mejor solución para todo tipo de tareas[20], ya que permite diseñar en el dispositivo reconfigurable los periféricos necesarios: coprocesadores, unidades operacionales, etc.

Como segunda alternativa se incluyendo una CPU dentro del dispositivo reconfigurable, lo que permite disponer de un sistema 100 % adaptable a la tarea a realizar: tamaño de memorias, potencia de la CPU, etc, y al mismo tiempo, añadir los periféricos necesarios. Esta flexibilidad permite, además, mejorar la conexión de los periféricos a la CPU de una forma directa y más óptima, eliminando la necesidad de implementar soluciones basadas en protocolos de comunicación serie o paralelo.

5.1 La CPU

La CPU diseñada presenta una arquitectura muy sencilla, con la intención de ir adaptándola a las necesidades que puedan surgir a lo largo del desarrollo del proyecto. Las características más relevante son:

- Arquitectura de 16 bits.
- Arquitectura Harvard, con memorias separadas para datos y programa.
- Arquitectura LOAD-STORE. Los accesos a la memoria de datos se harán sólo con estas instrucciones.
- Cuatro registros internos de propósito general.
- Cuatro registros internos de propósito general / indexación.
- Direccionamiento indexado.
- Un bit de *flag* para saltos condicionales.
- 8 instrucciones básicas: LOAD, STORE, MOVE, ADD, SUB, CMP, BF/BNF y JMP.

Para la implementación de la unidad de control se ha diseñado una máquina Mealy de tres estados: captura, decodificación/ejecución y almacenamiento. Las salidas de la máquina son las señales de control que se envían a los distintos elementos que la acompañan. Un diagrama de esta arquitectura se muestra en la figura 4. Además, se ha añadido un interface de comunicación con dos memorias de 256x16 bits, síncronas. La memoria de programa es una ROM, mientras que la de datos es SRAM, y ambas están implementadas dentro del dispositivo.

5.2 Componentes adicionales y aplicación

Para permitir que la CPU pueda controlar el robot se ha añadido al sistema formado por la CPU y las memorias, el conjunto de unidades *pwm* diseñado en 1. Los cuatro comparadores que forman parte de esta unidad tienen cada uno un registro asociado que es mapeado en la memoria de datos, de modo que el manejo de las unidades *PWM* es completamente transparente.

Para poder comparar esta alternativa con la presentada en 1, la CPU reproduce exactamente la misma secuencia de movimientos por lo que en la memoria de datos se almacena la tabla de control. El programa a ejecutar consiste en una rutina de lectura de las posiciones consecutivas de la memoria de datos y escritura de esos datos en las posiciones de memoria correspondientes a las unidades *pwm*.

Para el desarrollo del programa, y complementando al diseño VHDL de la CPU, se ha diseñado un ensamblador para esta arquitectura de modo que sea más sencilla su programación.

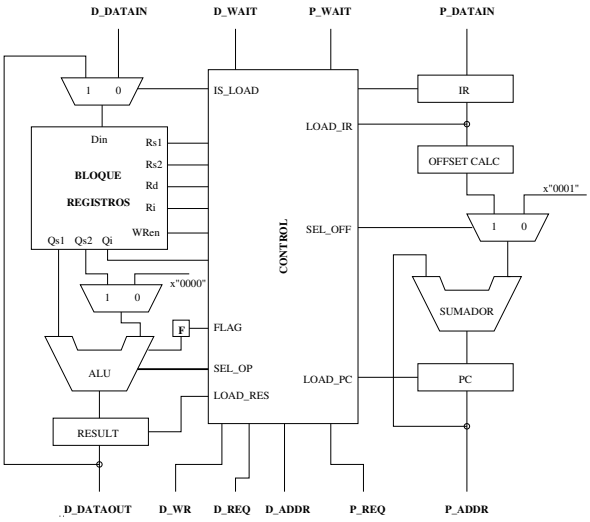

Fig. 4. Diagrama de bloques de la CPU (alternativa II).

6 Resultados y conclusiones

El diseño planteado como alternativa I presenta una ocupación de 78 CLBs, lo que supone un 39 % de la FPGA XS10 disponible en la placa JPS. La frecuencia máxima de operación no es crítica porque la velocidad de los servos es muy baja. Por otro lado, el sistema de CPU embebida planteado como alternativa II consiste en la CPU, la memoria con el programa de carga de secuencias, la memoria con las secuencias y las 4 unidades PWM, emplea 212 CLBs, por lo que no cabe en la FPGA disponible en la plataforma hardware. Es necesario utilizar dos placas JPS. Como mejor opción a la hora de particionar el diseño se ha planteado que la primera contenga la CPU y la memoria de programa y la segunda la memoria de datos y las unidades PWM, ya que estas últimas comparten con la memoria los buses de datos y dirección. Posteriormente, se ha planteado migrar el sistema a otra plataforma con más recursos: la placa Digilab2E, que posee una XC2S200E, cuya capacidad es suficiente para el diseño. Para este dispositivo, el sistema ocupa 285 slices (12 %) sin aprovechamiento de las BRAM.

La implementación del sistema de locomoción para *Cube Reloaded* usando una FPGA es viable. La alternativa I, basada exclusivamente en lógica combinacional y secuencial, consume un 63 % menos de recursos que la alternativa II, basada en una CPU empotrada. Es el diseñador el que puede elegir entre una u otra alternativa, según lo que prefiera: mayor flexibilidad o menor consumo de recursos. La utilización de microprocesadores/microcontroladores específicos, implica tener que adaptarse a las instrucciones y periféricos previstos por el fabricante. En tal caso, puede ser interesante utilizar una CPU empotrada y añadir el *hardware* estrictamente necesario, lo que ahorra recursos y facilita la programación.

Por otra parte, la robótica modular reconfigurable es la opción más versátil para la locomoción. Si añadimos la posibilidad de que los módulos estén constituidos por FPGA*s* en vez de por microprocesadores específicos, la versatilidad en el diseño es todavía mayor, pudiéndose implementar nuevas arquitecturas y permitiendo reconfiguración dinámica del *hardware* según las necesidades de locomoción, un campo todavía no explorado.

Agradecimientos

Este trabajo es financiado parcialmente por el Proyecto TIC 2001-2688-C03-03 del Ministerio de Ciencia y Tecnología de España.

Referencias

1. Robot Ambler en el CMU. http://www.ri.cmu.edu/projects/project_362.html
2. J. Bares, M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons, and W.L. Whittaker, "Ambler: An Autonomous Rover for Planetary Exploration," IEEE Computer, Vol. 22, No. 6, June, 1989, pp. 18-26. Disponible on-line en: http://www.ri.cmu.edu/pubs/pub_1431.html#abstract
3. Robot Dante II. http://www.ri.cmu.edu/projects/project_163.html
4. J. Bares and D. Wettergreen, "Dante II: Technical Description, Results and Lessons Learned," International Journal of Robotics Research, Vol. 18, No. 7, July, 1999, pp. 621-649.
5. Página personal de Mark Yim. http://robotics.stanford.edu/users/mark/
6. Mark Yim. “Locomotion with a unit-modular reconfigurable robot”. Stanfofd University. Dic,1994. Disponible on-line en http://www-db.stanford.edu/TR/CS-TR-95-1536.html
7. PolyPod. Página en Stanford. http://robotics.stanford.edu/people/mark/polypod.html
8. PolyPod. Página del Parc. http://www2.parc.com/spl/projects/modrobots/chain/polypod/index.html
9. Mark Yim, David G. Duff, Kimon D. Roufas,"PolyBot: a Modular Reconfigurable Robot", IEEE Intl. Conf. on Robotics and Automation (ICRA), San Francisco, CA, April 2000. Disponible on-line en http://www2.parc.com/spl/projects/modrobots/publications/index.html
10. PolyBot. Página del PARC. http://www2.parc.com/spl/projects/modrobots/chain/polybot/index.html
11. Tarjeta CT6811. http://www.iearobotics.com/proyectos/ct6811/ct6811.html
12. J. González,“Diseño y construcción de un robot articulado que emula modelos animales: aplicación a un gusano”. Proyecto fin de carrera. ETSIT-UPM. 2001.
13. Robot ápodo Cube 2.0. http://www.iearobotics.com/personal/juan/proyectos/cube-2.0/cube.html
14. J. Gonzalez, P. Haya, S. Lopez-Buerdo, and E. Boemo, "Tarjeta entrenadora para FPGA, basada en hardware abierto", Seminario Hispabot 2003, Alcalá de Henares, Mayo 2003.
15. Tarjeta JPS. http://www.iearobotics.com/personal/juan/doctorado/jps-xpc84/jps-xpc84.html
16. GHDL: Compilador/simulador libre de VHDL basado en el GCC. http://ghdl.free.fr/
17. Wine. Implementación de la API de Windows para X y Unix. http://www.winehq.com/
18. Proyecto Labobot. http://www.iearobotics.com/personal/juan/doctorado/labobot/labobot.html
19. El proyecto labobot en la web de Labomat. http://www.ii.uam.es/~laboweb/LabWeb/index.php3?seccion=1&pagina=5
20. XS40, XSP Board V1.4 User Manual, XESS Corp, Apex, NC, 1999. http://www.xess.com/
21. Tarejta Digitalb 2E. http://www.digilentinc.com/Catalog/digilab_2.html